AMENDMENTS TO THE CLAIMS

Claims 1-3, 16-17, 19, 25, 31-32, 35-36, 40-42 and 44 have been amended. The following is a complete listing of the claims, which replaces all previous versions and listings of the claims.

1. (currently amended) A device for implementing an inter-queue ordering mechanism between different queues of an interconnect of a computer system, comprising:

a first circular queue;

a second circular queue, the first and second circular queues adapted to have an ordering dependency between them, wherein entries in the second circular queue are not allowed to pass entries in the first circular queue;

a first counter; and

a second counter, the first and the second counters adapted to increment whenever an entry is enqueued to or dequeued from the first circular queue~~, respectively~~.

2. (currently amended) The device of claim 1, wherein ~~the~~ an entry in the second circular queue cannot be dequeued before ~~the~~ an entry that was placed earlier in the first circular queue is dequeued.

3. (currently amended) The device of claim 1, wherein ~~the~~ an entry in the second circular queue cannot be dequeued before ~~the~~ an entry that was placed earlier in the first circular queue is dequeued and then acknowledged as having been completed.

4. (original) The device of claim 1, wherein the dependency ensures that one of the first and second circular queues can proceed with an interconnect transaction as long as the second circular queue entries are not allowed to pass older entries of the first circular queue.

5. (original) The device of claim 1, wherein the first and second circular queues are comprised in a peripheral component interconnect (PCI) system.

6. (original) The device of claim 1, wherein the first and second circular queues are comprised in an order enforcement mechanism.

7. (original) The device of claim 1, wherein the first counter rolls over to zero only once before the second counter rolls over to zero.

8. (original) The device of claim 1, wherein the first and the second counters roll over to zero after reaching respective maximum values.

9. (original) The device of claim 8, further comprising a saved input pointer that points to an entry in the second circular queue whenever the first counter rolls over to zero.

10. (original) The device of claim 9, further comprising a saved output pointer that points to the same entry in the second circular queue until the saved input pointer is updated again.

11. (original) The device of claim 1, wherein the first and second circular queues are incorporated in a North Bridge of the computer system.

12. (original) The device of claim 11, wherein the North Bridge comprises a host-PCI bridge.

13. (original) The device of claim 1, wherein the first and second circular queues comprise first-in-first-out (FIFO) memory.

14. (original) The device of claim 13, wherein the first circular queue further comprises randomly accessed memory elements.

15. (original) The device of claim 13, wherein the FIFO memory comprises random access memory (RAM), flip-flops, or register files as a building block.

16. (currently amended) The device of claim 1, further comprising ~~aek~~ acknowledge bits in the first circular queue for maintaining the ordering dependency.

17. (currently amended) The device of claim 16, wherein the ~~aek~~ acknowledge bits comprise random access memory (RAM), flip-flops, or register files.

18. (original) The device of claim 1, further comprising other circular queues, wherein the second circular queue has a dependency on the other circular queues such that an

entry in the second circular queue cannot be dequeued unless requirements for dequeuing that entry are met with respect to every queue that the second circular queue depends on.

19. (currently amended) A computer system implementing an inter-queue ordering mechanism between different queues of an interconnect, comprising:

a central processing unit~~(s)~~;

core logic connected to the central processing unit~~(s)~~, the core logic including:

a first circular queue;

a second circular queue, the first and the second circular queues having an ordering dependency between them, wherein entries in the second circular queue are not allowed to pass entries in the first circular queue;

a first counter; and

a second counter, the first and the second counters adapted to increment whenever an entry is enqueued to or dequeued from the first circular queue~~, respectively~~.

20. (original) The computer system of claim 19, wherein the first and second circular queues comprise first-in-first-out (FIPO) memory.

21. (original) The computer system of claim 19, wherein the first circular queue further comprises randomly accessed memory elements.

22. (original) The computer system of claim 19, wherein the FIFO memory comprises random access memory (RAM), flip-flops, or register files as a building block.

23. (original) The computer system of claim 19, wherein an entry in the second circular queue cannot be dequeued before an entry that was placed earlier in the first circular queue is dequeued.

24. (original) The computer system of claim 19, wherein an entry in the second circular queue cannot be dequeued before an entry that was placed earlier in the first circular queue is dequeued and then acknowledged as having been completed.

25. (currently amended) The computer system of claim 24, further comprising a dequeue pointer, wherein ~~for acknowledges always coming back in order,~~ the dequeue pointer is advanced when an entry is dequeued from the first circular queue, and wherein the second counter is incremented when ~~the~~ an acknowledge signal is received.

26. (original) The computer system of claim 19, wherein the dependency ensures that one of the first and second circular queues can proceed with an interconnect transaction as long as the second circular queue entries are not allowed to pass older entries of the first circular queue.

27. (original) The computer system of claim 19, wherein the first and second circular queues are comprised in a peripheral component interconnect (PCI) system.

28. (original) The computer system of claim 19, wherein the first and second circular queues are comprised in an order enforcement mechanism.

29. (original) The computer system of claim 19, wherein the core logic comprises a North Bridge connected to the CPU, the North Bridge including the first and second circular queues.

30. (original) The computer system of claim 29, wherein the North Bridge comprises a host- to-PCI bridge.

31. (currently amended) The computer system of claim 19, further comprising ~~aek~~ acknowledge bits in the first circular queue for maintaining the ordering dependency.

32. (currently amended) The computer system of claim 31, wherein the ~~aek~~ acknowledge bits comprise random access memory (RAM), flip-flops, or register files.

33. (original) The computer system of claim 19, wherein the first counter rolls over to zero only once before the second counter rolls over to zero.

34. (original) The computer system of claim 19, further comprising other circular queues, wherein the second circular queue has the same ordering dependency on these other circular queues, and wherein dequeuing requirements of that ordering dependency must be satisfied relative to all the other queues before the second circular queue can be dequeued.

35. (currently amended) The computer system of claim 19, wherein the first and second circular queues ~~are adapted to operate when the queues and/or ordering dependency uses~~ use look-ahead ~~and/or~~ or pipelining techniques.

36. (currently amended) A method of implementing an inter-queue ordering mechanism between different queues of a computer system, the method comprising:

enqueuing transaction entries in a first circular queue in the computer system;

enqueuing additional transaction entries in a second circular queue in the computer system;

ordering the dequeuing of transaction entries in the first and the second circular queues, the ordering comprising preventing the additional transaction entries in the second queue from passing the transaction entries in the first circular queue; and

incrementing a first counter and a second counter whenever an entry is enqueued to or dequeued from the first circular queue~~, respectively~~.

37. (original) The method of claim 36, further comprising incrementing the first counter and rolling the first counter over to zero after a maximum value of the first counter is reached.

38. (original) The method of claim 36, further comprising incrementing the second counter and rolling the second counter over to zero after a maximum value of the second counter is reached.

39. (original) The method of claim 36, wherein the ordering comprises rolling the first counter over to zero only once before the second counter rolls over to zero.

40. (currently amended) A device for implementing an inter-queue transaction ordering mechanism of a computer system, comprising:

first queue means for holding transaction entries;

second queue means for holding other transaction entries;

means for ordering transaction handling between the first and the second queue means such that entries in the second queue means are not allowed to pass entries in the first queue means; and

means for incrementing a counting means whenever an entry is enqueued to or dequeued from the first ~~circular~~ queue means.

41. (currently amended) The device of claim 40, wherein the first and the second queue means ~~comprises~~ comprise first-in-first-out (FIFO) memory means for holding the transactions.

42. (currently amended) The device of claim 41, wherein the first ~~circular~~ queue means further comprises randomly accessed memory elements.

43. (original) The device of claim 41, wherein the FIFO memory means comprises random access memory (RAM), flip-flops, or register files as a building block.

44. (currently amended) The device of claim 40, wherein the counting means ~~for incrementing~~ comprises a first counter and a second counter, the first and the second counters adapted to increment whenever an entry is enqueued to or dequeued from the first ~~circular~~ queue means~~, respectively~~.